Sistemas Multiagentes en Ambientes Dinámicos: Planificación Continua mediante PDDL

Germán Braun Mario Moya Gerardo Parra

email: {germanbraun,moya.mario,gerardopar}@gmail.com

Grupo de Investigación en Lenguajes e Inteligencia Artificial
Departmento de Teoría de la Computación
Facultad de Informática
Universidad Nacional del Comahue
Buenos Aires 1400 - (8300)Neuquén - Argentina

Resumen

Este trabajo se desarrolla en el contexto del proyecto de investigación *Sistemas Multiagentes en Ambientes Dinámicos: Planificación, Razonamiento y Tecnologías del Lenguaje Natural*. Específicamente, en la línea Planificación, la temática que se está investigando es el desarrollo de una arquitectura para agentes que soporte tanto *control reactivo* como *deliberativo*, de forma tal que el agente pueda actuar de manera competente y efectiva en un ambiente real.

Uno de los objetivos de esta investigación es el intento de dotar a un agente inteligente de ambas capacidades. Esto brindará la posibilidad de elegir cuál sería la mejor forma de actuar frente un problema determinado.

Además, otro de los resultados esperados de nuestra investigación es la implementación de un traductor del lenguaje PDDL (o un subconjunto relevante de él) para la descripción de dominios de planificación. Esperamos que el framework de *planificación continua* desarrollado en el contexto de nuestro grupo de investigación pueda aprovechar las características que ofrece PDDL. Esto permitiría una rápida aplicación a cualquier problema definido en el lenguaje, y la posibilidad de comparar resultados de rendimiento con otras soluciones al mismo problema.

Palabras Clave: Agentes Inteligentes, Sistemas Multiagentes, Planificación, Planificación Continua.

Contexto

Este trabajo está parcialmente financiado por la Universidad Nacional del Comahue, en el contexto del proyecto de investigación *Sistemas Multiagentes en Ambientes Dinámicos: Planificación, Razonamiento y Tecnologías del Lenguaje Natural*. El proyecto de investigación tiene prevista una duración de tres años, ha comenzado en enero del 2010 y finaliza en diciembre de 2012.

1. Introducción

La representación de los problemas de planificación (estados, teoría de dominio o acciones disponibles, metas, etc.) debe posibilitar que los algoritmos de planificación utilicen eficientemente las ventajas de la estructura lógica del problema. La clave es encontrar un lenguaje que sea lo suficientemente expresivo para describir una amplia variedad de pro-

blemas, pero también lo suficientemente acotado para permitir que algoritmos eficientes operen sobre ellos.

Uno de los primeros lenguajes de representación y, seguramente, uno de los más referenciados en la literatura de Inteligencia Artificial ha sido STRIPS[2]. La representación STRIPS describe el estado inicial del mundo mediante un conjunto completo de literales básicos (*ground*) y las metas son definidas como una conjunción proposicional. La teoría de dominio, es decir, la descripción formal de las acciones disponibles para el agente, completa la descripción del problema de planificación.

En la representación STRIPS, cada acción es descripta por dos fórmulas: la fórmula de precondición y la de poscondición. Ambas están constituídas por una conjunción de literales y definen una función de transición de un mundo a otro. Una acción puede ser ejecutada en cualquier mundo w que satisfaga la fórmula de precondición. El resultado de ejecutar una acción en un mundo w es especificado tomando la descripción de w, adicionando cada literal de la poscondición de la acción y eliminando literales contradictorios.

STRIPS está basado en la idea de que algunas relaciones en el mundo no son afectadas por la ejecución de una acción. Estas restricciones (entre varias otras como tiempo atómico, no existen eventos exógenos, los efectos de las acciones son determinísticos, etc.) permiten trabajar con algoritmos de planificación más simples y eficientes, pero dificultan la descripción de problemas más complejos o de problemas reales.

La necesidad de un lenguaje con un poder expresivo mucho mayor que los existentes hasta el momento, impulsó, a fines de los 90's, el desarrollo del lenguaje de representación PDDL (*Planning Domain Definition Language*)[8]. En la actualidad, PDDL es soportado por muchos planificadores y, al igual que STRIPS, se basa en la suposición de mundo cerrado, permitiendo que la transformación de estados pueda ser calculada agregando o eliminando literales de la descripción del estado de partida. PDDL intenta expresar la "*física*" del dominio, es decir, cuáles son los predicados, qué acciones son posibles, cuál es la estructura de las acciones compuestas y cuáles son los efectos de las acciones.

El lenguaje está factorizado en un conjunto de características, llamadas *requerimientos*, y cada dominio definido usando PDDL debe declarar qué requerimientos asume. El requerimiento por defecto es STRIPS.

En la actualidad, PDDL es considerado el *standard de-facto* de los lenguajes de representación y mucho desarrollo se está aplicando sobre su especificación a fin de alcanzar nuevos objetivos tales como la comparación empírica de la performance de los planificadores y el desarrollo de un conjunto standard de problemas en notaciones comparables.

Existen distintas implementaciones que actúan como traductores desde el lenguaje PDDL hacia diferentes lenguajes destino. Por ejemplo, en [17], se ha implementado un traductor para SWI-Prolog utilizando DCG (*Definite Clause Grammar*) y limitado a un subconjunto de PDDL 3.0[3]. El subconjunto considerado para este traductor no incluye algunas características de PDDL tales como: restricciones, precondiciones negativas y disyuntivas, acciones con restricciones de duración, predicados derivados, preferencias, precondiciones universales, precondiciones existenciales y efectos condicionales.

Un enfoque diferente es abordado en [16] donde se considera una gramática ANTLR (*ANother Tool for Language Recognition*)[13] para un nuevo subconjunto de PDDL 3.0. La gramática ha sido diseñada para generar un traductor PDDL hacia código Java. Posteriormente, en [18], se ha modificado la gramática ANTLR original para generar código Python. Al igual que la anterior, ambas gramáticas excluyen algunas características relevantes de PDDL.

Otra alternativa es provista por la librería *PDDL4J*[14], desarrollada bajo licencia de software libre CeCILL[1]. Esta librería Java contiene un traductor para PDDL 3.0 y todas las clases necesarias para manipular sus conceptos. El propósito de PDDL4J es faci-

litar la implementación, en código Java, de planificadores basados en PDDL.

2. Líneas de Investigación y Desarrollo

El proyecto de investigación *Sistemas Multiagentes en Ambientes Dinámicos: Planificación, Razonamiento y Tecnologías del Lenguaje Natural* tiene varios objetivos generales. Por un lado, el de desarrollar conocimiento especializado en el área de Inteligencia Artificial Distribuida. Además, se estudian técnicas de representación de conocimiento y razonamiento, junto con métodos de planificación[4, 20] y tecnologías del lenguaje natural aplicadas al desarrollo de sistemas multiagentes.

Específicamente, en la línea Planificación, la temática que se está investigando es el desarrollo de una arquitectura para agentes que soporte tanto *control reactivo* como *deliberativo*, de forma tal que el agente pueda actuar de manera competente y efectiva en un ambiente real. Hanks y Firby [5] sugieren tratar de alcanzar un sutil equilibrio de estas dos estrategias: *deliberación* y *reacción*. La primera implica tomar todas las decisiones factibles en forma tan anticipada en el tiempo como sea posible. La segunda estrategia, *reacción*, consiste en demorar las decisiones que se tomen tanto como se pueda, actuando sólo en el último momento posible.

A simple vista, el primer enfoque parece perfectamente razonable. Un agente que puede pensar a futuro será capaz de considerar más opciones y, por lo tanto, con previsión, estar más informado para decidir qué acción tomar. Por otra parte, ya que la información sobre el futuro puede ser poco confiable y, en muchas situaciones del mundo real, difícil o incluso imposible de obtener, parece razonable, también, la alternativa de actuar en el último momento. Es más, sería razonable que ninguna de las dos políticas, pensar bien a futuro o actuar en el último momento, se ejecute con la exclusión de la otra.

Uno de los objetivos de esta investigación es el intento de dotar a un agente inteligente de ambas capacidades. Esto brindará la posibilidad de elegir cuál sería la mejor forma de actuar frente un problema determinado.

Las capacidades deliberativas se logran a partir de la implementación de un planificador novedoso, denominado *planificación continua* [12], una de las alternativas para planificación en ambientes reales planteadas en [15]. En esta aproximación, se presenta un agente que persiste indefinidamente en un entorno, posiblemente cambiante y dinámico. Tal agente no se detiene al alcanzar un meta determinada, sino que sigue ejecutándose en una serie de fases que se repiten e incluyen la formulación de metas, planificar y actuar. Para ganar eficiencia y tiempo de deliberación, la arquitectura provee una *librería de planes* prediseñados por el programador del agente para que sean adaptados o reparados, para aplicarlos a situaciones particulares. Cada miembro de esta librería consiste de un *cuerpo* y una *condición de invocación*, indicando bajo qué circunstancias se puede aplicar este plan.

Asimismo, se tiene previsto que el diseño del agente de esta investigación tenga dos modos de operación: *reactivo* o *planificador*. Con estos dos modos, básicamente, se plantea un *subsistema de control* con dos posibles configuraciones. En la primera, el planificador tiene el control por defecto y sólo cuando no pueda resolver una determinada situación, le transmite el control al modo reactivo. En la otra posible configuración, el modo reactivo está a cargo y le pasa el control al modo planificador en situaciones previamente identificadas por el diseñador del agente. Este subsistema se implementa como un conjunto de *reglas de control*. Estas reglas de control permiten determinar cuál de los modos de operación tendrá el control del agente en determinada situación.

3. Resultados Obtenidos y Esperados

La arquitectura de control basada en planificación continua se encuentra en estado de desarrollo. Algunos resultados de esta investigación han sido publicados en [12, 9].

El caso de estudio en que ha sido aplicada la arquitectura de control es el fútbol con robots. El control reactivo para este problema se encuentra desarrollado bajo el nombre de *Rakiduam* [6, 10].

Rakiduam es un equipo de fútbol de robots con licencia GNU (General Public License) que ha participado en numerosas ediciones del Campeonato Argentino de Fútbol con Robots (CAFR) con resultados más que satisfactorios[7, 6, 11, 19].

La irrupción de PDDL como standard de lenguaje de representación genera nuevos desafíos motivados en la necesidad de desarrollar y/o extender herramientas que lo soporten y la ampliación de su expresividad para adecuarlo a los dominios de aplicación destino.

Uno de los resultados esperados de nuestra investigación es la implementación de un traductor del lenguaje PDDL (o un subconjunto relevante de él) para la descripción de los dominios y de las acciones, de manera tal que puedan ser manipuladas por el framework de *planificación continua*[12]. Es esperable que el framework pueda aprovechar las características que ofrece PDDL. Esto permitiría una rápida aplicación a cualquier problema definido en el lenguaje, y la posibilidad de comparar resultados de rendimiento con otras soluciones al mismo problema.

4. Formación de Recursos Humanos

El actual proyecto de investigación es una continuación de la línea de investigación abierta en el proyecto anterior: *Técnicas de Inteligencia Computacional para el Diseño e Implementación de Sistemas Multiagentes.*

A partir de las líneas de trabajo planteadas en el actual proyecto de investigación, se tratará de dar inicio a, por lo menos, dos nuevas tesis de Licenciatura en Ciencias de la Computación.

Además, se espera el inicio de la consolidación como investigadores de los miembros más recientes del grupo.

Referencias

[1] CeCILL. Website. http://www.cecill.info/index.en.html.

[2] R. Fikes and N. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *J. Artificial Intelligence*, 2:189–208, 1971.

[3] A. Gerevini and D. Long. BNF Description of PDDL 3.0. 2005.

[4] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning. Theory and Practice*. Morgan Kaufmann, 2004.

[5] S. Hanks and R. J. Firby. Issues in architectures for planning and execution. In *Workshop on Innovative Approaches to Planning, Scheduling and Control*, pages 59–70, Scheduling and Control, San Diego, CA, November 1990.

[6] P. Kogan, M. Moya, G. Torres, J. Yañez, L. Cecchi, G. Parra, R. del Castillo, and C. Vaucheret. RAKIDUAM: Un Equipo de Fútbol con Licencia GNU General Public License. In *V Campeonato Argentino de Fútbol con Robots*, 2007.

[7] P. Kogan, J. Yañez, C. Campagnon, L. Cecchi, G. Parra, C. Vaucheret, and R. D. Castillo. Aspectos de diseño y de implementación del equipo de fútbol con robots RAKIDUAM. In I. U. A., editor, *Proceedings del Campeonato Argentino de Fútbol con Robots (CAFR)*, 2006.

[8] D. McDermott. PDDL, the Planning Domain Definition Language. Technical

report, Yale Center for Computational Vision and Control, 1998.

[9] M. Moya, P. Kogan, G. Parra, L. Cecchi, and C. Vaucheret. Sistemas multiagentes en ambientes dinámicos: Planificación. In *XII Workshop de Investigadores en Ciencias de la Computación*, El Calafate, Santa Cruz, Argentina, 2010. Universidad Nacional de la Patagonia San Juan Bosco.

[10] M. Moya, G. Torres, P. Kogan, C. Vaucheret, R. del Castillo, L. Cecchi, and G. Parra. Framework para el desarrollo de equipos de fútbol de robots. In *VI Campeonato Argentino de Fútbol con Robots*, Buenos Aires 1400, Neuquén, Argentina, 2008. Universidad Nacional del Comahue.

[11] M. Moya, G. Torres, P. Kogan, C. Vaucheret, R. del Castillo, L. Cecchi, and G. Parra. Framework para el desarrollo de Equipos de Fútbol de Robots. Caso de Uso: RAKIDUAM. In *V Workshop de Inteligencia Artificial aplicada a la Robótica Móvil*, pages 58–62, Universidad Nacional del Comahue - Neuquén - Argentina, 2008.

[12] M. Moya and C. Vaucheret. Agentes deliberativos basados en planificación continua. In *X Workshop Agentes y Sistemas Inteligentes (WASI)*, Martiarena esquina Italia - S.S. de Jujuy, Octubre 2009. Universidad Nacional de Jujuy - Facultad de Ingeniería.

[13] T. Parr. Antlr v3. Website. http://www.antlr.org.

[14] D. Pellier. Pddl4j. Website. http://www.math-info.univ-paris5.fr/~pellier/doku.php?id=research:software.

[15] S. Russell and P. Norvig. *Artificial Intelligence: A modern approach*. Prentice Hall, New Jersey, third edition, 2009.

[16] Z. Saigol. Pddl grammar for antlr v3. Website. http://www.murat-knecht.de/projekte/aspplan/dist/Pddl.g.

[17] R. Sasak. Planner prolog. Website. http://artax.karlin.mff.cuni.cz/~sasar5am/pddl.

[18] Schuerfen. Antlr v3 grammar for pddl with python as target. Website. http://schuerfen.blogspot.com/2009/05/antlr-v3-grammar-for-pddl-with-python.html.

[19] D. Trevisani, G. Braun, M. Moya, R. del Castillo, L. Cecchi, G. Parra, P. Kogan, C. Vaucheret, and G. Torres. RAKIDUAM: sistema multiagentes para el fútbol de robots. In *VII Campeonato Argentino de Fútbol con Robots*, Cabildo 134, Morón, Buenos Aires, 2009. Facultad de Informática, Ciencia de la Comunicación y Técnicas Especiales de la Universidad de Morón.

[20] J. Zhang, X. Nguyen, and R. Kowalczyk. Graph-based Multi-agent Replanning Algorithm. In *Proceedings of the Sixth Intl. Joint Conf. on Autonomous Agents and Multiagent Systems*, 2007.